Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Technologies

Bachelor Thesis

Social Network: The Affect Of a Social Network Specialized For University.

Anar MAMMADHASANOV

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

Anar Mammadhasanov

Informatics

Thesis title

Social Network: The Affect Of a Social Network Specialized For University.

Objectives of thesis

The aim of this thesis is to create a social media platform within the university committee using the React framework.It is aimed to eliminate the lack of social communication that arose during the pandemic period, and at the same time, the university will facilitate internal information exchange data.As a result of the survey conducted among students at our University, it is found that the different activities that the university has registered for students are not very effective and important by mail.For this reason, the project I am developing offers the points I am aiming for and the problems it will solve:

1.It is aimed that new students will be more adapted to the university environment.
2.It is aimed that university students will find themselves more useful as a part of the university.
3.The communication between the student and the teacher at the university is more convenient and on a comfortable platform for both parties.
4.Informing students about social events and important projects at the University in a more interactive and attractive way
5.It is aimed to bring together administrative staff, teachers and students on the same platform and deconstruct the communication problems that normally exist in the network.

Methodology

The methodology of the thesis, the desired methodology will be discussed in 3 sections.The first part will be made on the User Interface and will be made in the standard non-JSX format using HTML and module CSS inside the Javascript code without reaching the business logic layer, and the first part will be completed in this way.

The second part will be created as a Connect installation.

The Connect abundance is intended to play a bridge role between the BLL and the UI, as well as ensure that users cannot access the BLL part of the UI and ensure security.

The final part of the thesis will consist of BLL.The BLL is intended to be installed via Redux.Reducers will be created for all sections on the social media platform that is planned to be built, and we plan to create reducers from inicial state, functions and their action creators, and finally, the reducers will be collected in redux and created in the store.

The proposed extent of the thesis

30-40

Keywords

React,Redux, Flux Conception,Social Network

Recommended information sources

Josef Pavlíček, Cookbook of interaction design for HCI and ID, Online material: https://docs.google.com/presentation/d/1nbLjgEX5mS6kl_cRx6CeKuhd-fzz-kyYn_j03vMLkH4/edit?usp=sharing

Expected date of thesis defence

2021/22 SS – FEM

The Bachelor Thesis Supervisor

Ing. Josef Pavlíček, Ph.D.

Supervising department

Department of Information Engineering

Electronic approval: 7. 2. 2022

Ing. Martin Pelikán, Ph.D.

Head of department

Electronic approval: 21. 2. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Dean

Prague on 12. 03. 2024

Declaration

I declare that I have worked on my bachelor thesis titled "Social Network: The Affect Of a Social Network Specialized For University." by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break any copyrights.

In Prague on 10.03.2024 ______________________________

Acknowledgement

Hereby I would like to thank all those who supported me in preparing my bachelor's thesis. I would especially like to thank my supervisor doc. Ing. Josef Pavlicek, Ph.D for his professional guidance, assistance, and valuable comments throughout this work. Last but not least, I would like to thank to my family and collegues from different development teams for their material and spiritual support during my bachelor studies.

Sociální síť: Vliv sociálního inženýrství na kybernetické útoky - rizika, zranitelnosti a protiopatření

Abstrakt

Tato bakalářská práce se zabývá studiem a tvorbou specializované sociální sítě, která by dokázala řešit všechny unikátní výzvy, kterým čelí univerzity během pandemie. Název této práce je "Sociální síť: Efekt specializované sociální sítě pro univerzity", a bude se podrobně věnovat zkušenostem souvisejícím s administrativou, vzděláváním a sociálními problémy, které byly zesíleny náhlým přesunem výuky online. Pandemie změnila způsob, jakým studenti učí a komunikují mezi sebou, stejně jako s učiteli a administrativními orgány. Proto by mělo existovat digitální řešení, které by mohlo překlenout komunikační mezery a zachovat to, co znamená univerzitní život.

Autor tohoto projektu chce vytvořit digitální centrum pro studenty, učitele, administrativní personál a všechny ostatní zapojené do univerzitního života. Bude obsahovat funkce jako sítění (ve stylu sociálních médií), aktualizace událostí, sdílení akademických zdrojů, stejně jako příležitosti pro stáže nebo práci - vše speciálně určené pro univerzity. Tato síť si klade za cíl znovu vytvořit pocit, že jste na skutečné univerzitě, ale z pohodlí vašeho domova.

Pokud jde o softwarové návrhové vzory, tato práce využívá React pro uživatelské rozhraní, protože je to v současnosti jedno z nejpopulárnějších frameworků, známé svou intuitivností a jednoduchostí. Redux je použit pro správu vrstvy obchodní logiky, zatímco Axios pokrývá úkoly komunikace se serverem. Všechny tyto technologie dohromady vytvářejí platformu, která bude fungovat rychle, aniž by někoho nechala v rozpacích.

Tento výzkum by mohl dát vzniknout něčemu opravdu užitečnému v dobách, jako jsou tyto, kdy vzdělávací instituce si nemohou dovolit další uzavření kvůli nepředvídaným okolnostem jako je COVID-19. Mícháním technické inovace s porozuměním tomu, co lidé potřebují z uživatelské perspektivy, by tento komplexní seznam mohl pomoci ještě více zlepšit vzdělávací zapojení a komunikaci, než jak tomu bylo před rokem 2020.

Klíčová slova: COVID-19, vzdělávací technologie, sociální síť, univerzitní komunikace, React, Redux, Axios, online vzdělávání, reakce na pandemii, zapojení studentů.

Social Network: The Affect Of a Social Network Specialized For University

Abstract

This bachelor's thesis is about the studying and creation of a specialized social network that would be able to handle all the unique challenges universities are facing during the pandemic. The title of this thesis is "Social Network: The Effect Of A Social Network Specialized For University", it will dive deep into the experiences when it comes to administration, education, and social issues that have been amplified by moving classes online so suddenly. The pandemic has changed how students learn, and communicate with each other, as well as with teachers and administrative bodies. That's why there should be a digital solution that could bridge gaps in communication and preserve what university life means.

The author of this project wants to make a digital hub for students, teachers, administration staff, and everyone else involved in university life. It will have features like networking (social media style), event updates, academic resource sharing as well as opportunities for internships or jobs — all made especially for universities. This network aims to recreate the feeling of being at an actual university on your home computer.

In terms of software design patterns, this thesis uses React for user interface because it's one of the most popular frameworks at the moment known for its intuitiveness and simplicity. Redux is used to handle business logic layer management whilst Axios covers server communication tasks. All those technologies combined create a platform that will work fast without leaving anyone scratching their heads.

This research might give birth to something truly useful in times like these where educational institutions can't afford another lockdown due to unforeseen circumstances like COVID-19. By blending technical innovation with an understanding of what people need from a user perspective, this comprehensive list could help enhance educational engagement and communication even more than what we've had before 2020.

Keywords: COVID-19, educational technology, social network, university communication, React, Redux, Axios, online learning, pandemic response, student engagement

Table of content

1. Introduction

Information technology has penetrated into the field of education. Distance learning using Internet technologies is a form of education, along with full-time and part-time. But for the first time, educational institutions around the World are faced with the problem of a "pandemic". The root of the problem is divided into two parts. The first part is an educational institution that was not planned to transfer all studies for everyone online. All teachers had to drastically change the entire methodology of study and the principle of work. In addition, the system of administrative work was not ready for such changes. The other side is the students. Students who initially chose online study did not change anything for them. But the students who go to university, for them have also changed from the roots. For them, not only the principle of learning has changed, but also the relationship with teachers. The other side of the problem is the social side, which affects as well as the change in the principle of study. According to my own research, students who are transferred to distant from face-to-face study also move away from study, as the atmosphere at the university creates the feeling that students feel they are part of the university.The purpose of this thesis is to explore this new problematic phenomenon and find the right solution for students in advancing their studies.

The new generation will require new solutions. The basis of this thesis is the creation of a new social network, which will combine maintaining communication between students-students, students-teachers, students-administration.

In this thesis, we made an attempt to understand what we need to know and be able to create a social network for the university, what software is a tool for creating a social network and how to use it effectively. We will build our project using the React framework. In addition, we will develop business logic layer (BLL) using Redux and using a real server and by using Axios (connection instrument), we will develop a data access layer (DAL)

As a result, we get an application in which all the nodes of the problem are connected. This application contains part of the modern solution to this problem, and by developing the server part we can fully solve many communication problems both during and after the pandemic.

2 Objectives and Methodology

2.1 Objectives

The aim of this thesis is to create a social media platform within the university committee using the React framework.It is aimed to eliminate the lack of social communication that arose during the pandemic period, and at the same time, the university will facilitate internal information exchange data.As a result of the survey conducted among students at our University, it is found that the different activities that the university has registered for students are not very effective and important by mail.For this reason, the project I am developing offers the points I am aiming for and the problems it will solve:

1.It is aimed that new students will be more adapted to the university environment.
2.It is aimed that university students will find themselves more useful as a part of the university.
3.The communication between the student and the teacher at the university is more convenient and on a comfortable platform for both parties.
4.Informing students about social events and important projects at the University in a more interactive and attractive way
5.It is aimed to bring together administrative staff, teachers and students on the same platform and deconstruct the communication problems that normally exist in the network.

2.2 Methodology

The methodology of the thesis, the desired methodology will be discussed in 3 sections.The first part will be made on the User Interface and will be made in the standard non-JSX format using HTML and module CSS inside the Javascript code without reaching the business logic layer, and the first part will be completed in this way.

The second part will be created as a Connect installation.
The Connect abundance is intended to play a bridge role between the BLL and the UI, as well as ensure that users cannot access the BLL part of the UI and ensure security.

The final part of the thesis will consist of BLL.The BLL is intended to be installed via Redux.Reducers will be created for all sections on the social media platform that is planned to be built, and we plan to create reducers from inicial state, functions and their action creators, and finally, the reducers will be collected in redux and created in the store.

3 Literature Review

3.1 Introduction to the literature review

The rapid infusion of information technology into education, accelerated by the COVID-19 pandemic's push for e-learning has opened up new boundaries in educational technology.This review tackles the role of software design patterns in improving digital learning environments with a focus on university-centered social network development. At its core is the Model-View-Controller (MVC) framework. MVC is a fundamental design pattern that divides application development into three interconnecting sections: Model for data management, View for user interface, and Controller for user input and system output [1].Despite its usefulness over time, complex web applications have exposed the limitations of MVC in terms of scalability. This has prompted the hunt for more lively patterns (Figure 1)

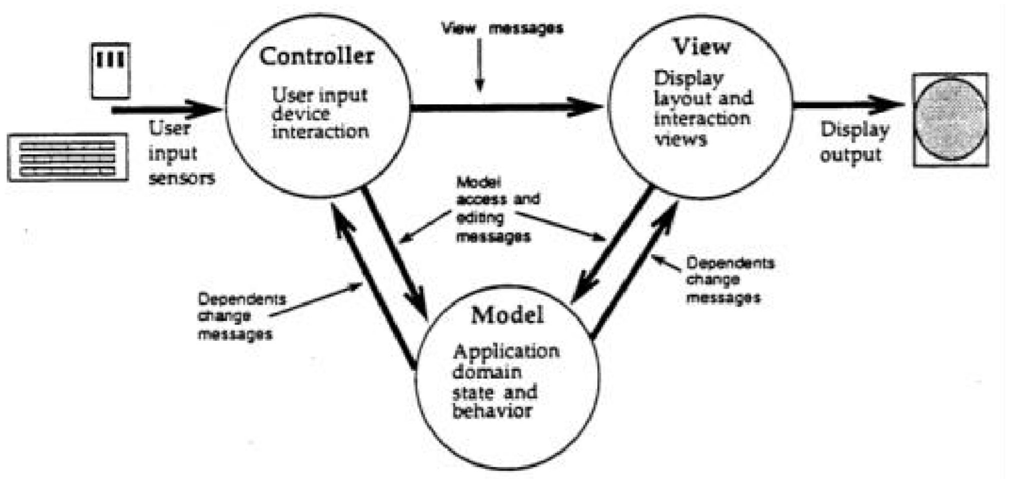

Controller
User input device interaction
View messages
View
Display layout and interaction views
Display output
User input sensors
Model access and editing messages
Dependents change messages
Dependents change messages
Model
Application domain state and behavior

Figure 1. Model-View-Controller State and Message Sending

The Flux architecture has been posited as a solution to these limitations, proposing a unidirectional data flow and a more centralized control of application state, which has proven effective for large-scale applications like those utilized by Facebook [2] (Figure 2). This review will investigate how Flux, and by extension Redux (Figure 3), offer predictable state management for JavaScript applications, a vital feature for responsive web applications [3][4].

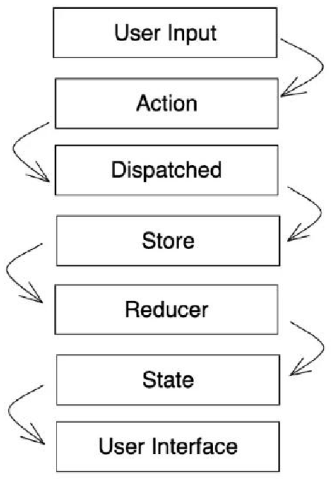


Figure 2. The Flux architecture,unidirectional data flow

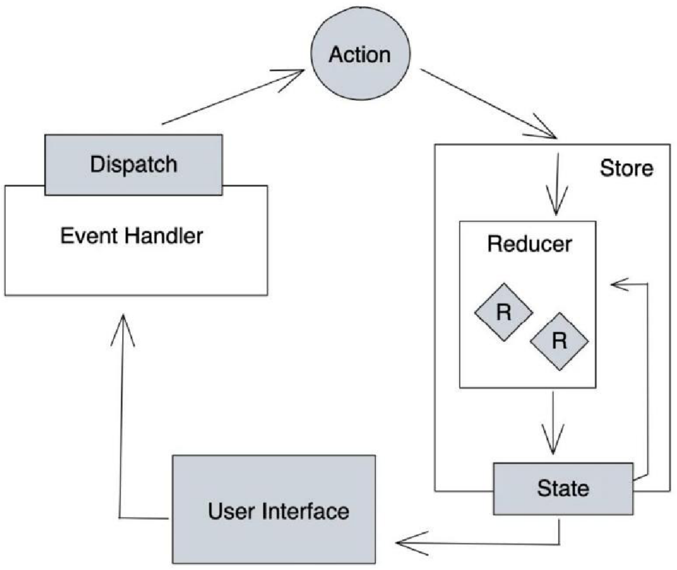


Figure 3. Redux architecture

ReactJS has influenced the development of user interfaces a lot through its component-based approach. It offers a declarative way to program which has made it so that developers can make dynamic and responsive web pages much easier [3]. With the help of Redux and Axios, ReactJS makes up the main structure for modern-day web applications that need real-time data handling and seamless user experience[4][5].

By methodically analyzing the literature, this review will evaluate just how much potential these technologies have when it comes to aiding in the development of a social network platform. A platform that not only caters to university communities but also helps contribute to enriching our educational experience in this day and age where we rely heavily on technology for education.

3.2 Software Design Patterns

Software design patterns are like the blueprints builders use when they're constructing a house. They are not the actual walls or roof, but the plans that guide how these parts come together to make a building that stands strong and serves its purpose well. Donald Le Vie Jr [1] points out that these patterns help people who make software explain their complex ideas more clearly, much like a well-drawn map can help travelers find their way.

Greg Perry [2] takes this idea to beginners in programming, showing them how these blueprints can help create software that's not just a jumble of code, but a well-organized machine that's easy to understand and fix when things go wrong.

Researchers like Nazar, Aleti, and Zheng [3] take it a step further by developing ways to automatically spot these patterns in software, ensuring that the right blueprint is used at the right time, which is crucial for a software's speed and ability to grow without breaking.

Christopher Alexander [4], who first talked about the idea of patterns in building design, showed how the same good design ideas could work in different places and for different purposes, and that's a concept that's very useful in software, too.

Krasner and Pope [5] focused on a specific design pattern called MVC — short for Model View-Controller. Imagine you're watching a play: the Model is the story happening behind the scenes, the View is the stage where you see the story unfold, and the Controller is like the director, making sure everything happens when and how it's supposed to.

These patterns are tools that software developers use every day. They help them make apps and programs in a way that other developers can easily understand and work on too. By reading the work of smart people like Depew [6] on how social media fits into learning, Boduch [7] on the nuts and bolts of Flux architecture, and Abel [8] on the nitty-gritty of making apps with React, developers can see these blueprints in action.

In the bustling world of web development, books and articles [9-18] are the trusty guides that show developers how to use patterns to build websites and apps that are easy to use, quick to load, and simple to maintain. Whether it's the basics of HTML from TutorialsPoint [14], the ins and outs of JavaScript from Zakas [15], or the cutting-edge techniques of React and Redux from Chinnathambi [17] and Gelman and Dinkevich [18], there's a wealth of knowledge out there.

3.2.1 Model-View-Controller (MVC)

The Model-View-Controller (MVC) framework is a software design pattern and one of the most basic paradigms for organizing and structuring web applications. The application is divided into three interconnected parts, where each part does different things to make it work in harmony with other parts while being independent. One of these parts is responsible for gathering data from the user while another one's job is to show data that was collected to the user.

- **Model:** This part represents the core data and business logic of the application, making it possible to store, retrieve, and process data efficiently. It has all the rules and operations related to an application's data responsibilities.
- **View:** With a name that just makes sense intuitively, this part creates what users see on their screens as they interact with an app or website, etc. It presents features from the model in a way that's easy for people to understand and use.
- **Controller:** The third part sits between the first two acting like a bridge that connects them, making it possible to do all the tasks necessary with ease and flow from start to end with no interruptions. This part handles requests from users who want something specific done within your program, manipulates data using model functions, and then selects a view based on what needs to be shown back to users among many other things. It is responsible for directing traffic effectively so you can get whatever you need to be done faster.

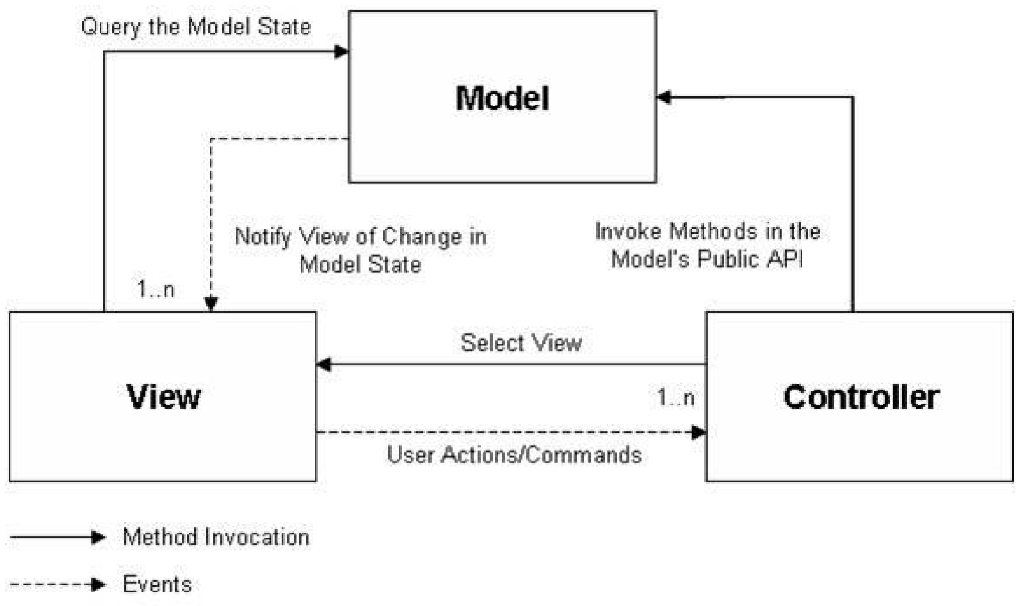


Figure 4. Model-View-Controller (MVC)

Advantages of MVC

- **Dividing and conquering:** This approach splits the application into a few clearly defined parts, which makes it easier to manage and keeps code development organized.
- **Easy to grow:** The modular design lets you scale and develop any part of an app without much hassle.
- **Take two, they're small**: Often components can be reused for different parts of the same app, which speeds things up considerably.

Challenges of MVC

- **Building a labyrinth:** As apps get larger, the number of interdependencies between various components in MVC grows alarmingly fast, so keeping those in check can get challenging.
- **Slow speed ahead:** When views and models are connected, sometimes that may lead to performance issues, particularly when it comes to handling data-intensive applications.
- **Scaling mountains:** Let's say you're building complex systems; managing the data flow between different parts can quickly become a nightmare, and scaling only makes things more complicated.

Application of MVC

Frameworks like Ruby on Rails, ASP.NET MVC, and Spring MVC are widely used in web development and make the best examples of MVC design patterns. This model is especially useful when there is a need for different data presentations or where the same data-set is accessed from different presentation templates.

3.2.2 Redux Architecture

Redux is an open-source JavaScript library used to manage application state mostly used in conjunction with React or Angular libraries to build user interfaces (Figure 5). It serves as a predictable state container that provides consistent behavior across client, server, and native environments so that applications can be easily tested.

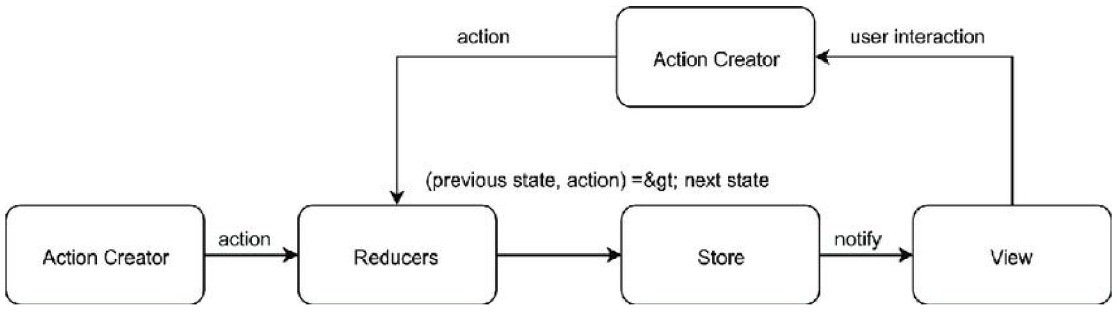


Figure 5. Redux structure

- Actions: Actions are payloads of information that send data from the application to the Redux store. They are sent using store.dispatch() and they are plain JavaScript objects having a type property indicating what kind of action is being performed.

- Reducers: Reducers specify how the application's state changes in response to actions sent to the store. They are pure functions that take the previous state and an action and return the next state. They are called reducers because they're the type of functions you would pass to **Array.prototype.reduce(reducer, initialValue)**

- Store: Actions and reducers are brought together by the store, which stores the application state, exposes the state through getState(), updates the state with dispatch(action), and registers listeners via subscribe(listener). It also handles the unregistering of listeners via the function returned by subscribe(listener).

- View: The view in Redux is usually a React component that responds to state changes. It calls on the required state from the Redux store and then renders itself so it can show the new data.

Advantages of Redux

- Predictable State Management: Actions in Redux can only change the present application’s state.
- Maintainability: By centralizing the application's state, there is coherence when developing code thereby making it easy to manage.
- Developer Tools: In terms of developer debugging and checking how states change over time, Redux presents robust tools.

Challenges of Redux

- Complexity: For small projects, Redux can introduce unnecessary complexity and overhead.
- Verbosity: There could be cases where much boilerplate code is written thus appearing to be redundant, especially in the case of simple tasks.
- Steep Learning Curve: New developers find it hard to understand the principles and structure of Redux as it takes some time for familiarization.

Application of Redux

In complex web applications that handle a lot of states spread across multiple areas in an application, this framework is commonly employed. The framework best suits systems where managing states directly affects how stable an app becomes such as those for enterprises or data-driven interfaces a lot more heavily towards them.

3.3 Data Flow in Application Design

Application design data flow enables the movement of information through various application components. As an example, it determines how well users interact with the system or how fast it responds and is designed in the application as well as influences system interactions. The choice of a data flow model has significant implications for the design and scalability of an application.

3.3.1 Bidirectional Data Flow in MVC

One important feature of the Model-View-Controller (MVC) framework is a bidirectional data flow that supports interactive user experiences. This allows quick transfer of data from one part to another part of the MVC, which in turn facilitates dynamic updates and real-time synchronization.

Dynamics: In addition, MVC architecture lets View display data from Model, while user's interactions with View can initiate changes on the Model side too. Also when users make some inputs these are considered by the Controller that makes corresponding amends within Model, and then back to View this happens.

Advantages:

- **Interactive Experience:** It improves responsiveness and makes a system more engaging by enabling instant updates between UI and Data Model.
- **Real-Time Synchronization:** It maintains synchronization between UI and underlying data so that users can always work with up-to-date information.
- **Adaptive UI Development:** It helps create flexible & intuitive interfaces responsive to changeable data & user input.

Challenges:

- **Management Complexity:** Complexity may rise rapidly because multiple elements are closely knit into a single architecture especially when they become large-scale applications using the MVC approach.
- **Circular Dependencies:** There will be heavy dependencies on debugging or maintenance if the model has tight coupling with the view resulting in circular dependencies.
- **Performance Overheads:** This means continuous synchronization of data hence adverse performance may be experienced mostly where there is much manipulation of data happening.

Contextual Use

Bidirectional data flow is especially helpful to applications that demand repeated updating and user interaction like dynamic web apps, and real-time analytics platforms.

3.3.2 Unidirectional Data Flow in Redux

Redux introduces a unidirectional data flow model that simplifies state management and enhances predictability within applications, particularly those built with React.

Dynamics: The unidirectional flow in Redux kicks off when the view dispatches actions, which are then processed by reducers to update the state before reflecting it to the view.

Advantages:

- **Predictable State Changes:** This creates a well-defined path for state updates thus making system behaviors predictable.
- **Debugging Simplicity:** Due to having one direction of data flow and clear state transitions, debugging becomes straightforward.
- **Testing Efficiency:** The straight-line nature of how Redux handles data makes it easier to do tests and guess outcomes.

Challenges:

- **Setup Complexity:** Setting up Redux for the first time can be cumbersome and overwhelming for novice developers.
- **Boilerplate Code:** This might lead to verbosity in code which may be unnecessary for simple applications.

Contextual Use

In cases where maintaining state consistency matters most such as in large-scale single-page applications (SPAs) with heavy state management requirements, the uni-directional flow of redux is shining.

3.4 Redux vs. MVC: A Comprehensive Analysis

3.4.1 Enhanced State Management and Predictability

- **Centralized State Management:** With Redux, the state of an entire application is stored in a single object; this is the opposite approach that MVC takes [7][17]. Centralizing everything makes it easy to manipulate state and also improves predictability– which can be important when dealing with complex applications and managing state through multiple components [17]. Meanwhile, MVC's way of managing state across multiple components just leads to inconsistencies and more debugging.
- **Predictable Data Flow:** With predictable data flow comes a clear path that data takes while passing through your app using Redux's architecture [10][16]. Not only does this make everything easier to understand, but also much simpler when it comes to tracking state changes [10][16]. Easier debugging means easier maintenance, preventing potential headaches down the line [10][16]. On the other hand, MVC patterns tend to struggle with this level of simplicity due to their bidirectional data flow between components — which leads to more complex interactions overall.

3.4.2 Scalability and Maintainability

- **Modular Code Structure:** Every reducer in Redux has its responsibility when it comes to handling state changes [11][17]. This alone brings on a code structure that is so organized and scalable because each reducer focuses on its specific part of the state — making it less confusing overall if you ask us [11][17]. So not only does this make development faster by dividing work easily among members, but also makes updating/scale-ups smoother [11][17].

- Ease of Integration: The integration between Redux and modern UI frameworks at their core is what helps with scalability/maintainability [8][13]. Using both of these technologies allows devs to leverage their strengths — efficient state management of Redux and powerful UI capabilities of frameworks like React [8][13]. As amazing as MVC frameworks are, they don't have this level of seamless integration with modern front-end technologies. For that reason, it may not be the best choice when complex and modern web applications arise.

3.4.3 Debugging and Testing Efficiency

- Simplified Debugging Process: Redux's structure is like a single-engine train, it only goes one way and has one centralized station for managing state systems, causing the debugging process to be generally less complex than others. With Redux, developers can easily see how and when states change, this makes it faster to trace back to the root of the bug. This isn't the case for MVC architectures with their spaghetti data flows intertwined together, making it difficult to find where bugs originate from, especially in larger-scale applications.

- Enhanced Testing Capabilities: The organization of Redux allows for simpler and more efficient testing techniques compared to other frameworks or libraries out there. Since side effects are isolated, it's easier to write unit tests and integration tests rather than having interdependent models, views, and controllers like most other frameworks.

3.4.4 Integration with Modern Development Practices

- Compatibility with Component-Based Frameworks: Redux is fully compatible with component-based frameworks such as React. Having compatibility like this gives developers a lot more time to build out reusable components that manage their state/behaviors since they don't have to worry too much about trying not to break anything

in the overarching system while doing so. MVC patterns may not offer the same level of synergy with component-based frameworks which could lead to less efficient development workflows.
- **Community and Ecosystem Support:** In terms of user support/community size for Redux vs MVC framework vs just any regular library out there (most likely including yours), Redux has strengths over many others due to its strong community base providing infinite resources such as middleware extensions (which are great for handling side effects), debugging tools developed by creators, and more libraries that have been released by the community. The MVC framework also has a strong community but it may lack the specific focus and breadth of resources that Redux offers. Especially when looking at state management solutions.

3.4.5 Complex User Interface and State Management

- **Handling Complex States:** If you have an application that has a lot of moving parts, then Redux might be the best tool for you to use. Because of its architecture, managing complicated state transformations is way easier within the application. In big applications where managing states can get out of hand, using MVC could end up backfiring due to the potential for increased complexity. Plus components can become tightly coupled.
- **Flexible and Dynamic UI Development:** With Redux being built into frameworks like React, developers can create very dynamic interfaces that respond to user inputs promptly. When you compare it with MVC, there's no guarantee that you'll be able to develop at the same speed or even offer the same level of flexibility and ease of development, especially if you're working on something with complex UI requirements.

3.4.6 Adaptability and Performance Optimization

- **Adapting to Changing Requirements:** Designing an application that's easy to modify is key when things start changing regularly. Because Redux developers are able to easily add or modify features without impacting anything else within the overall structure of their app, this makes it perfect for modern web development. If you're using an MVC framework,

changes might have a bigger impact on your codebase since they're usually more rigid and less flexible than Redux architectures.
- **Performance Considerations:** When it comes down to tweaking performance, large datasets can slow down an application. One thing we know about Redux though is that its efficient state management optimizes performance by eliminating unnecessary re-renders. With typical MVC architectures that update frequently and sync data between views and models, performance can suffer especially when you're dealing with complex data synchronization between the two.

3.4.7 Redux vs. MVC: Elevating State Management in Web Development

As a front-end developer, I've been fascinated with Redux and its bidirectional data flow versus the traditional MVC pattern and unidirectional data flow. Through my journey on this subject, I've learned so much about how to use Redux in modern web application development. Let's start with the state management capabilities of both systems. The centralized and predictable state management system of Redux is a significant improvement over MVC's distributed state management, especially when it comes to manipulating states and tracing changes — which are both vital for the scalability and maintainability of applications.

Next up is the modular code structure enabled by Redux's reducer-based architecture. It makes it easier to organize, scale, and maintain the codebase — effectively addressing any scalability challenges you might come across in MVC frameworks.

Moving forward is debugging and testing efficiency. The unidirectional data flow of Redux simplifies debugging, reducing time spent on identification and resolution of issues; while the centralized state management significantly enhances testing capabilities. Integration with modern development practices such as React has uncovered new efficiencies with component-based frameworks that no one knew existed before now.

Complex user interface states can also be easily managed using Redux without much hassle. This flexibility allows for rapid changes without refactoring extensive parts of your app — saving you plenty of time in the fast-paced environment web development has become today.

Lastly, Redux has impressive performance optimization capabilities, especially in complex cases. Works best when handling complex state logic and large datasets. It's an approach that efficiently manages state, reducing unnecessary re-renders thus optimizing application performance.

In conclusion, this research has convinced me that Redux is the go-to choice for modern web applications. Especially when compared to MVC patterns and other unidirectional data flow patterns. Redux excelled in four key areas: scalability, debugging, integration with modern development practices, and most importantly, state management. This makes it superior to MVC when it comes to contemporary web development scenarios.

3.5 Comparative Analysis of React, Angular, and Vue.js Frameworks

In the world of web development, any developer will tell you that the technology stack is everything. It's the most important factor in determining if an app will be a success or not. This segment of the report will showcase three key technologies in web development: React, Angular, and Vue.js.

React was developed by Facebook and is known for its virtual DOM feature. It helps optimize rendering and boosts performance. On the other hand, we have Angular – Google's creation which boasts a full-fledged MVC framework with two-way data binding and a suite of features right out of the box.

Finally, there's Vue.js, a progressive framework that combines features from both React and Angular for flexibility and simplicity during application development.

Let's dive into their architectural paradigms to examine how each framework handles components, state management, and user interface rendering. After that, we shall look at the necessary technicalities of these frameworks, such as React's JSX syntax, Angular's dependency injection, and Vue's reactive data binding. The evaluation also looks at how easy it is to integrate with other tools and libraries as well as scalability in large-scale applications and community support all of which are important factors in modern web development.

Similarly, this analysis will study the impact of these technologies on web development. This includes things like the speed of development, new developer ramp-up time required for new developers, different project sizes fit, and quick adaptability to changing needs in web development.

Thus, through a comprehensive comparative study, this section seeks to equip web developers and decision-makers with invaluable insight to help them choose the most suitable technology that aligns with their project objectives and long-term strategy.

3.5.1 React Framework

React is a JavaScript library that Facebook developed to make interactive interfaces. The biggest draw for developers is the reusability of components, which makes solving complex problems and building dynamic, data-driven web applications possible.

Technical Abilities and Structure

- **Component-Based Architecture:** Each component in React has its state and logic, which allows them to be combined into complex user interfaces. By breaking down every piece into its smallest parts, developers can reuse code more effectively and keep the project maintainable.
- **Virtual DOM (Document Object Model):** The in-memory copy of the real DOM is called "virtual." This feature lets React optimize how it manipulates HTML without sacrificing speed or performance in large-scale applications.
- **JSX (JavaScript XML):** With JSX syntax extension, writing code that looks like HTML with JavaScript behind the scenes becomes possible. This makes the UI readable and simplifies how components are structured.
- **Unidirectional Data Flow:** Developers don't always have to pass around data, but when they do with React, it must be done in one direction only. Child components can't modify their parent's state directly; instead, data changes are communicated through a system of "props."
- **React Hooks:** For functional components that are lightweight and performant, Hooks lets developers use stateful logic without converting them into classes.

- **Lifecycle:** Components go through different stages as they exist within an application – these stages are called its lifecycle methods. They provide developers opportunities to update UIs and application states at exactly the right time.

Mounting

- **`constructor()`:** The first technique to be employed in the process of initializing state and binding event handlers.
- **`render()`:** Noted to be invoked, a method used to render the component into the DOM.
- **`componentDidMount()`:** Used for the sole purpose of being called immediately after the component is stuck into the DOM. Its main uses are for network requests, DOM manipulations, and subscriptions.
- **Updating (Figure 6):**
    - **`shouldComponentUpdate()`:** Determines whether the component should be re-rendered. It's used for performance optimizations.
    - **`render()`:** Called again to re-render the component if shouldComponentUpdate returns true.
    - **`componentDidUpdate()`:** Invoked after the component's updates are flushed to the DOM. Useful for handling post-render operations.
- **Unmounting:**
    - **`componentWillUnmount()`:** Called before the component is removed from the DOM. It's used for cleanup activities like invalidating timers and canceling network requests.

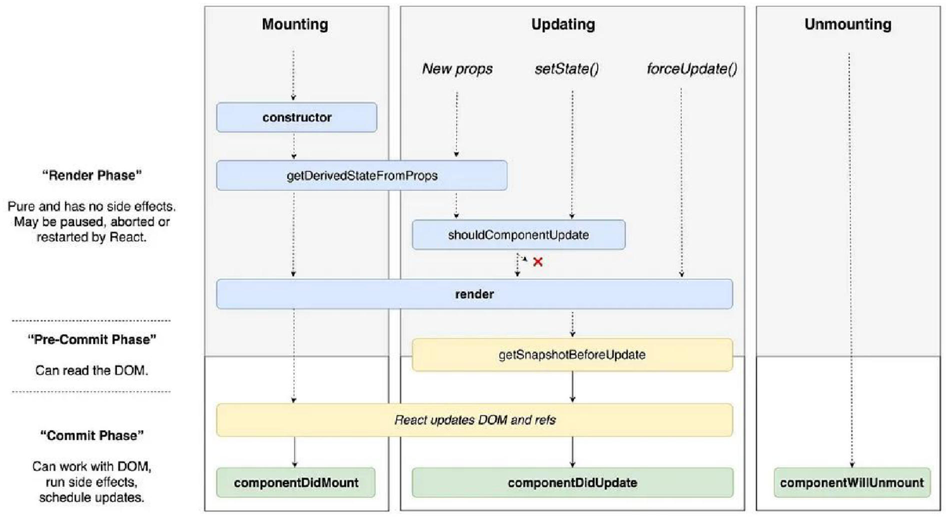


Figure 6. React life cycle structure

3.5.2 Angular Framework

Angular, which Google has developed and continues to maintain, is a strong, all-inclusive framework with an MVC architecture that's used to build dynamic web apps. It's known for being durable and powerful, as well as for including a lot of tools that make it capable of building enterprise-level applications.

Technical Abilities and Structure

- **Component-Based Architecture:** This framework structures every one of its applications around components, which are what make up the user interface. Each one comes with a TypeScript class, an HTML template, and styles that are associated with them.
- **TypeScript:**TypeScript is the building block of Angular - it's based on JavaScript offers developers optional static typing (a type checker), as well as classes and interfaces so you can catch errors before they happen. It enhances code predictability and quality while also making it easier to find and fix problems when they come up later on.
- **Modules:** Modules are self-contained blocks of related functions that are put together in order to form the entire program. Depending on how many there are in an app, each module could define different features like services or pipes.
- **Dependency Injection:** Thanks to this aspect of Angular's setup, objects and services can be more effectively created and managed within the framework's programs. This makes it easier to build larger apps without forcing too much extra work onto their creators.
- **Two-Way Data Binding:** When some data changes in either the model or view side of things in Angular's apps, this feature automatically ensures that those changes are reflected on both sides immediately afterward.
- **Directives:** Developers who want to take more control over elements' behavior in their programs' DOMs could turn toward directives —which extend the HTML vocabulary— in order to do so effectively.
- **Services and Dependency Injection:** Services are single, reusable objects that can package together important business logic and data services. Thanks to Angular's dependency injection systems, they're easily plugged into different components throughout the app.

- **RxJS and Observables:** Reactive programming using observables is handled by the RxJS library, which Angular has integrated within its system. This is useful for managing the state of different parts of an application, as well as handling HTTP requests in a way that's more flexible than what other libraries or frameworks could offer on their own.

Advantages

- **All-in-One Solution:** Angular is often called a 'batteries-included' framework, offering solutions that are normally not included like routing, state management, forms, etc..
- **TypeScript's Strong Typing:** TypeScript integration brings object-oriented features, better tools, and error checking so it makes the code more maintainable.
- **Very Performant:** Thanks to Angular's ahead-of-time (AOT) compiler, change detection, and dependency injection big applications will run generally fast.
- **Lots of Tools and Ecosystems:** Angular CLI is a command-line interface tool that simplifies project scaffolding. It also supports development and testing while having strong support from the community.

Disadvantages

- **Steep Learning Curve:** With its comprehensive design and array of concepts it becomes harder to learn than simpler frameworks, especially for beginners or those coming from less feature-rich frameworks.
- **Too wordy and complex:** According to other developers Angular has too many words to be verbose and too complicated with strict architecture TypeScript's extensive use of decorators and dependency injection.
- **Performance in Large Applications:** While Angular performs well on its own making complex and large applications may result in performance issues if not optimized properly, especially regarding change detection.

In conclusion, Angular offers powerful features that make it hard for other frameworks to compete with it at an enterprise level. This covers all the bases though learning it might be time-consuming because of how much you have to learn about the framework itself let alone coding with it properly, It can get pretty confusing with the way it's built but it tries its best to catch any compiling errors at runtime which makes up for how difficult it can get, And if you know what you're doing after spending countless hours learning this beast then this is definitely one of the best choices you could make when working on a big project with complex requirements.

3.5.3 Vue.js Framework

A progressive JavaScript framework used for creating user interfaces, Vue.js is designed to be incrementally adoptable. That means you can use it as a small part of your project without causing any integration issues with other JavaScript libraries. Conversely, you can use it as the main tool to build Single-Page Applications if you're willing to use some modern tooling and supporting libraries.

Technical Abilities and Structure:

- **Reactive and Composable View Layer:** This sets the focus on the view layer right away by offering reactive data binding and composable view components, making interactive and dynamic user interfaces a piece of cake.
- **Single-File Components:** By including template, script, and style sections all in one file, these components are easily readable and maintainable. It also gives them a self-contained nature.
- **Reactivity System:** Using a simple API, Vue's reactivity system tracks changes made to the application state so that it can automatically update the DOM when updates are made.
- **Virtual DOM:** Borrowed from React, Vue.js uses its own virtual DOM which allows it to optimize rendering in complex applications while still keeping direct manipulation of the regular DOM at a minimum. The result is improved performance.
- **Template Syntax:** Declaratively binding the rendered DOM to an underlying component's data is easy thanks to Vue.js' HTML-based template syntax. This simple approach makes development more intuitive for those who know HTML well enough.
- **Ease of Integration:** Implementing this framework into your existing projects will not be difficult or time-consuming work. Additionally, using up-to-date tools with this framework in complex single-page applications (such as Vue Router or Vuex) will yield great results too.

- **Transition System:** Vue presents a solid system for adding transition effects to elements when they come and go from the DOM. This makes it easy to add some extra flair to the user interface.

Advantages

- **Ease of Learning:** Vue's gentle learning curve is one of its most significant benefits. It allows people with basic knowledge of HTML, CSS, and JavaScript to get started quickly.
- **Flexibility and Simplicity:** Despite its simplicity, Vue is very flexible. It can be used in many different projects, no matter how small or large they may be.
- **High Performance:** Because it's lightweight and utilizes a virtual DOM, Vue.js performs well, especially with complex user interfaces.
- **Robust Community and Ecosystem:** Even though it's relatively new, Vue has a fast-growing community that includes several libraries and tools for easier use.

Disadvantages

- **Risk of Over-flexibility:** Being so flexible might come back to bite you if you're working on larger teams or more complex projects; things might start going off the rails without you even realizing it until it's too late.
- **Smaller Community Compared to React and Angular:** As much as the community continues to grow at a rapid pace, Vue still doesn't have as large of a community compared to other similar frameworks like React and Angular; this could result in fewer resources and third-party integrations being available.
- **Perceived as Less Suitable for Very Large-Scale Projects:** People will always say what they want about something new entering the scene. In this case, because Vue came later than other frameworks like React did (and also has less widespread support), some people think it's just not built right for very large-scale applications.

In summary, developers love using Vue.js because of how well it works altogether. It's great as an easy-to-learn framework for smaller projects or as a more complex one for large applications. All this stuff it carries around with it ensures an efficient and straightforward experience.

3.5.4 Conclusion

After evaluating React, Angular, and Vue.js, I conclude that React is the optimal choice for my social network application. Analyzing their strengths, weaknesses, and ability to meet specific platform demands led me to this decision.

Why React Is the Top Pick

- **Component-Driven Development:** Having a similar structure to a social network application strengthens its modular UI elements and reusability-speeding up development time and easing maintenance.
- **Performance Optimization:** Real-time user interaction requires a dynamic system capable of smooth updates, so performance is paramount in choosing a framework. React's virtual DOM handles it with ease.
- **Vast Ecosystem and Community Support:** With an array of libraries and tools, React's ecosystem will help me add advanced features to the social network.
- **Scalability and Flexibility:**** React's potential to scale combined with its ease of integration makes it perfect for a long-term project expected to grow over time.

Redux's Role in My Decision

Redux is a state management tool that greatly complements the capabilities of React, especially when it comes to handling the complex state of a social network application.

By centralizing the store, Redux ensures that state management is done consistently and predictably throughout the application. This makes the state-handling code much simpler to write and easier to understand, even with numerous user interactions and real-time data updates. The predictability and maintainability offered by Redux play a key role in managing the flow of data within a social networking platform.

Additionally, Redux's ease of debugging and its ability to support middleware for asynchronous operations make it an excellent match for building a solid, interactive social network app.

Final Thoughts

Considering all this information, React combined with Redux seems like my best choice for developing this social network application. I believe this pairing will help me create an app that’s both high-performing and easy to scale and maintain as social networking needs evolve. This decision represents a strategic approach to using the best tools available to build an engaging and efficient platform for people to interact.

4 Practical Part

This project’s purpose is to develop a unique social network that will effectively handle communication for university communities. The platform aims to meet the needs of schools in times like this pandemic, by supporting educational engagement and connectivity. It also allows people to share resources easily and engage with one another. That being said, it should foster efficient information exchange, community building, and collaborative learning. By doing so, the network caters to both students and faculty/staff alike for a tailored environment that promotes productive academic collaboration as well as healthy social interaction.

4.1 Design and Architecture

Akin to a chameleon, our social network system is adaptable to universities' needs. To be user-friendly, accessible, and efficient, our design approach has ensured a smooth experience from the beginning.

The User Interface (UI) is simple but effective. We prioritize getting users where they need to be with ease. From registration to profile management, communication tools to content sharing, it’s all done within a couple of clicks. By using minimalistic design principles we reduce clutter and enhance usability. The UI works just as well on your phone as it does on your computer or tablet.
To make this happen we had to integrate several core components:

- **User Registration and Profile Management:** This one's self-explanatory but worth mentioning that users have privacy settings available so they can customize their experience (Figure 7, Figure 8).

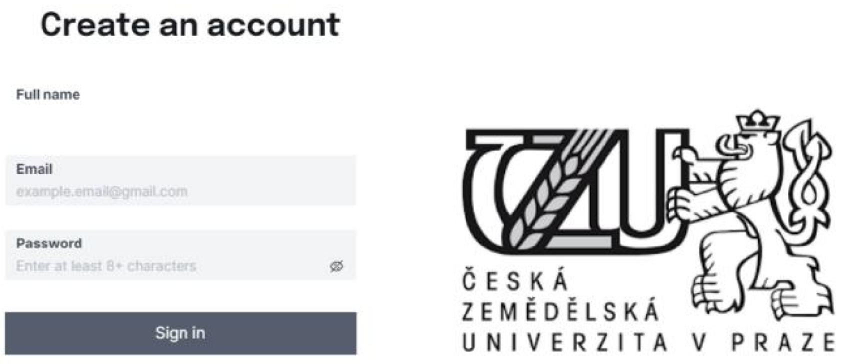


Figure 7. Registration page

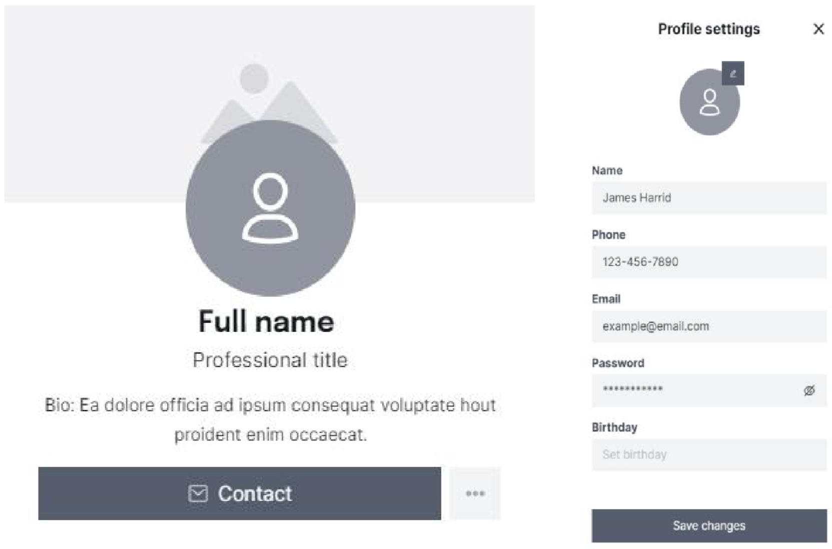
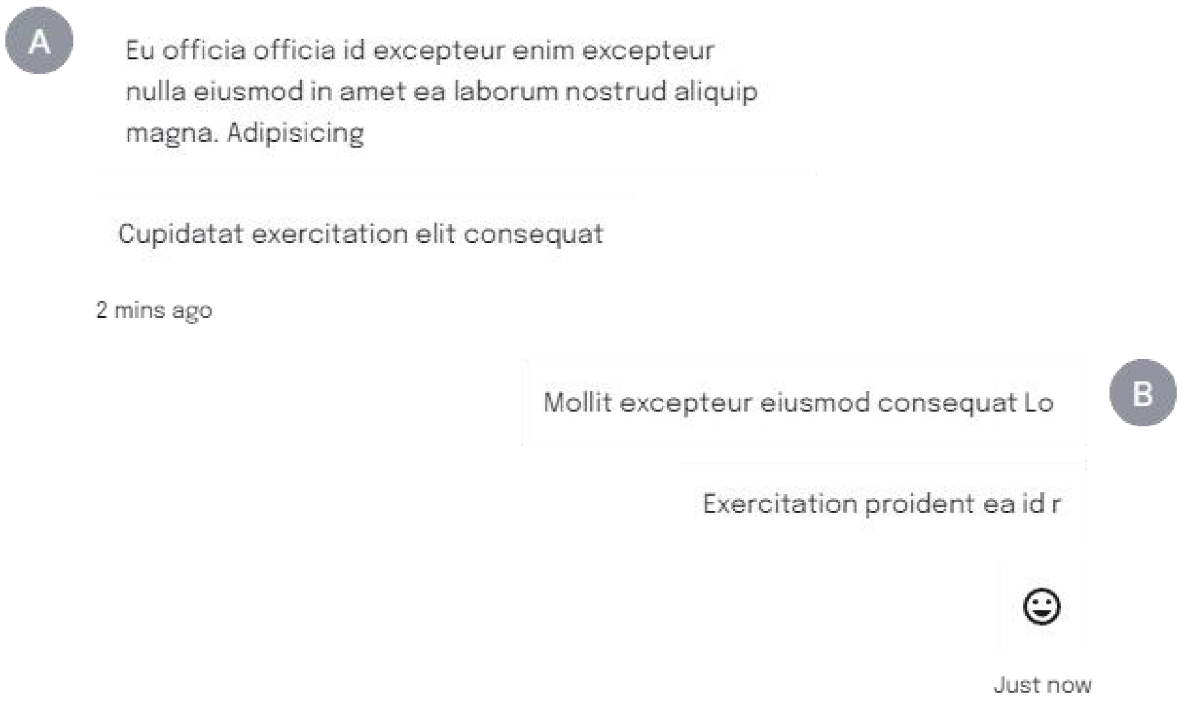


Figure 8. Profile page details

- **Communication Tools:** As much as we like independent thinking there's nothing quite like collaboration. Our platform takes direct messaging to its most convenient level by including forums and group chats as well (Figure 9).


A

Eu officia officia id excepteur enim excepteur nulla eiusmod in amet ea laborum nostrud aliquip magna. Adipisicing

Cupidatat exercitation elit consequat

2 mins ago

Mollit excepteur eiusmod consequat Lo

B

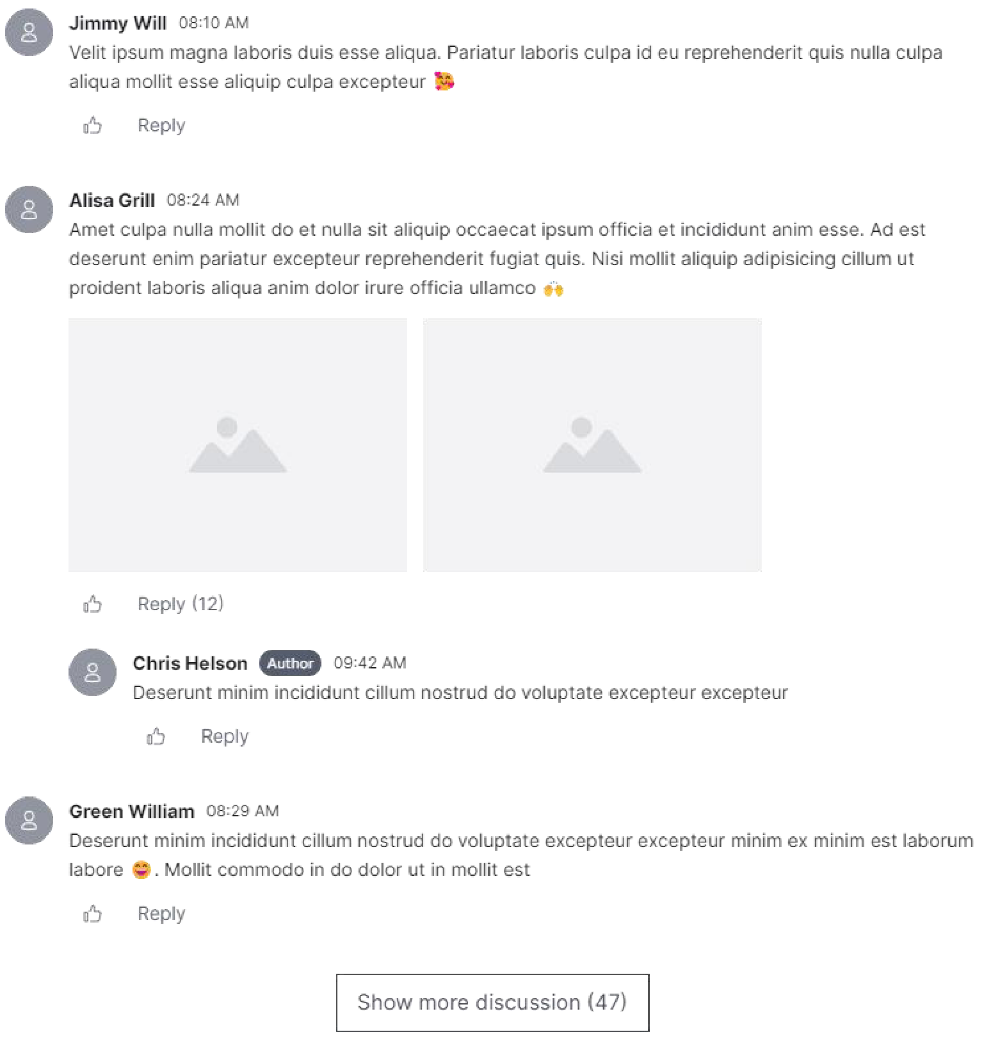
Exercitation proident ea id r

☺

Just now

Figure 9. Personal chat design

- **Content Sharing and Academic Collaboration:** Sharing documents can always be a headache but not here. Alongside class schedules and academic resources, students will have no problem spreading their knowledge around (Figure 10).


Jimmy Will 08:10 AM
Velit ipsum magna laboris duis esse aliqua. Pariatur laboris culpa id eu reprehenderit quis nulla culpa aliqua mollit esse aliquip culpa excepteur 🥰

Reply

Alisa Grill 08:24 AM
Amet culpa nulla mollit do et nulla sit aliquip occaecat ipsum officia et incididunt anim esse. Ad est deserunt enim pariatur excepteur reprehenderit fugiat quis. Nisi mollit aliquip adipisicing cillum ut proident laboris aliqua anim dolor irure officia ullamco 🙌

Reply (12)

Chris Helson Author 09:42 AM
Deserunt minim incididunt cillum nostrud do voluptate excepteur excepteur

Reply

Green William 08:29 AM
Deserunt minim incididunt cillum nostrud do voluptate excepteur excepteur minim ex minim est laborum labore 😄. Mollit commodo in do dolor ut in mollit est

Reply

Show more discussion (47)

Figure 10. Group chat for discussion

- **Data Management and Security:** With great power comes great responsibility which is why we hold data integrity and confidentiality in high regard. We employ advanced security measures to protect personal information.

This architecture supports the network's goal of enhancing communication, sharing resources, and improving engagement while also providing a scalable solution for the growing needs of university communities everywhere.

4.2 Technology Stack

The technological foundation of the social network is crafted to offer robustness, high performance, and seamless user experience, adhering to the latest industry standards and best practices. The selection of each technology was strategic, aimed at addressing specific needs within the platform while ensuring adaptability and future growth.

4.2.1 Development Technologies

Frontend Technologies:

- **React:** This JavaScript library is central to the development of the user interface, chosen for its declarative and efficient approach to building complex user interfaces. React's virtual DOM optimizes rendering, improving performance and user experience. The component-based architecture enhances code reusability and maintainability.
- **Redux:** Serving as the state management tool, Redux complements React by providing a centralized application state, making it easier to manage state across the entire application. This is particularly beneficial for complex features like user authentication, content management, and real-time notifications
- **Sass** (Syntactically Awesome Style Sheets): An extension of CSS that enables more advanced features like variables, nesting, and inheritance, making the styling more systematic and easier to manage. Sass helps in writing clean, maintainable styles for a consistent look and feel across the platform.

Backend Technologies:

- **Node.js:** Selected for its non-blocking, event-driven architecture, Node.js facilitates efficient data processing and high concurrency, essential for handling multiple simultaneous user requests without slowing down.
- **Express.js:** A minimal and flexible Node.js web application framework that provides a robust set of features to develop APIs and web applications. It simplifies the server-side logic and routing, improving the development speed and quality.

Database Management:

- **MongoDB (NoSQL) :**A document-oriented NoSQL database, chosen for its flexibility in handling large volumes of structured and unstructured data. Its dynamic schema nature allows for rapid iterations and modifications without the need to redesign the database structure.

4.2.2 APIs & Middleware

- **Axios:** A JavaScript library for making HTTP requests from the browser to the backend services, Axios has some advantages over other libraries when it comes to its wide browser compatibility and ability to intercept request and response data.
- **Passport.js:** An authentication middleware for Node.js that simplifies the process of authenticating users with different strategies like OAuth and local authentication, Passport is known for being easy to use, yet still secure. Plus, it works well with various user authentication methods.

4.2.3 Deployment and Version Control:

- **Git:** Used by developers around the world, Git allows multiple people to work on the same project without stepping on each other's toes. It also helps track changes so if something goes wrong it's easy to revert.
- **Docker:** Docker is a platform that allows you to containerize your application. This ensures consistency across different environments such as development, testing, or production so there are fewer bugs.
- **Heroku:** Cloud platforms normally make things easier in terms of deployment and scaling applications like this one. By integrating with Git repositories it allows teams devs easy continuous delivery so they can focus more time elsewhere.

The comprehensive integration of these technologies provides a solid foundation for the social network platform, ensuring scalability, security, and a high-quality user experience. Each component of the technology stack was chosen not only for its individual merits but also for its compatibility and synergy with other elements, fostering a cohesive and efficient development environment.

4.2.4 Development and Build Tools

- **Webpack:** It's a module bundler that takes your assets (JavaScript, CSS, images) and creates a bundle you can include on your website. This ultimately reduces load times and optimizes performance.
- **Babel:** A compiler mainly used to convert ECMAScript 2015+ code into a backward-compatible version of JavaScript. This way, you can write modern JavaScript features but still have your app work across all browsers.

4.3 Development Process

The process of development for the social network can be broken down into various different stages, each designed to take the company closer towards their finished product. This approach allows a clear path from idea to reality while also maximizing efficiency.

4.3.1 Planning and Analysis

- In this beginning phase, information is gathered about what the university community needs.
- Key features are identified and scoped out.
- Existing solutions by competitors are studied to see if there's something we can do differently or better than them.
- Objectives are defined with clear timelines and resources allocated accordingly.

4.3.2 Design

- Wireframes and design mockups are made to give a visual representation of what users will see when they're on the site.
- A system is established consisting of color schemes, typography, and UI elements that keep the look consistent across all pages.

- User flow diagrams will be used to help designers understand how people will navigate through the platform in order to build an easy-to-use website.

4.3.3 Development

- Setting up the development environment and configuring the technology stack.
- The development is divided into manageable sprints following the Agile methodology, allowing for incremental progress and regular assessment.
- Frontend and backend development proceed in parallel, with continuous integration and collaboration between teams.
- Implementation of core features such as user registration, content management, and communication tools, following the best coding practices and standards.

4.3.4 Testing

Comprehensive testing phase including unit tests, integration tests, and end-to-end tests to ensure the functionality and performance of the platform.

Conducting usability testing with potential users to gather feedback and identify areas for improvement.

Addressing bugs and issues that arise during testing promptly to maintain the development timeline.

4.3.5 Deployment

Preparing the platform for deployment, including final optimization and security checks.

Utilizing continuous integration and continuous deployment (CI/CD) pipelines for streamlined deployment to the production environment.

Monitoring the deployment process to ensure successful launch and operation of the platform.

4.3.6 Post-Deployment and Maintenance

Post-launch phase involves continuous monitoring of the platform for performance, security, and user feedback.

Regular updates and enhancements based on user feedback and technological advancements.

Implementing a support system to address user queries and issues effectively.

This structured development process ensures that the platform is developed with a user-centric approach, meets the identified needs, and adheres to high standards of quality and performance. By following this methodology, the project aims to deliver a comprehensive and effective social network for the university community.

4.4 Implementation Details

The implementation of the social network platform is meticulously planned to ensure the delivery of a user-friendly, efficient, and secure environment. Below are insights into the implementation of key features:

4.4.1 User Registration

Implemented a streamlined registration process requiring minimal user input to enhance user experience.

Integrated email verification to ensure authenticity and prevent spam accounts.

Used Passport.js for authentication, supporting both local authentication methods and social logins for user convenience.

Encryption of passwords using bcrypt to ensure user data security.

4.4.2 Profile Management

Developed a user profile module allowing users to customize their profiles, including personal information, educational background, and areas of interest.

Implemented privacy settings enabling users to control the visibility of their information.

Provided the ability to upload and change profile pictures, enhancing personalization and user identity.

4.4.3 Communication Tools

Developed an integrated messaging system enabling private and group conversations, fostering community engagement and collaboration.

Implemented real-time notifications using WebSockets to alert users about new messages and updates.

Included community forums and discussion boards for sharing information, ideas, and academic resources.

4.4.4 Content Sharing

Enabled users to post, share, and edit content, including text, images, and videos, supporting academic and social collaboration.

Implemented a tagging system to categorize content and facilitate easy search and discovery.

Incorporated like and comment functionalities to promote interaction and feedback among users.

4.4.5 Security and Privacy

Ensured the platform's security by implementing SSL/TLS for encrypted data transmission.

Conducted regular security assessments and audits to identify and mitigate vulnerabilities.

Established clear data privacy policies and adhered to GDPR standards to protect user information.

4.4.6 Performance and Scalability

Optimized front-end performance using React's virtual DOM to minimize page load times.

Utilized MongoDB's efficient data retrieval and indexing features to handle large volumes of data.

Implemented scalable server architecture to accommodate growing numbers of users and data.

4.4.7 User Interface and Experience

Adopted a mobile-first design approach to ensure accessibility and usability across various devices.

Conducted A/B testing to refine UI elements and workflows, ensuring intuitive navigation and user interactions.

By focusing on these implementation details, the social network platform aims to provide a comprehensive, secure, and engaging environment for the university community, promoting connectivity, collaboration, and academic success.

4.5 Scalability and Performance

- Challenge: Making sure our platform can support a lot more users and data without slowing down.
- Solution: Used a setup where different tasks are handled by separate services (microservices) and cloud solutions like AWS Elastic Beanstalk and MongoDB Atlas to automatically adjust resources as needed.

4.5.1 User Data Security and Privacy

- Challenge: Keeping user information safe and private as online security threats grow.
- Solution: Added strong security steps like HTTPS, data encryption, and safe login methods. Regularly checked our security to keep up with rules on protecting data.

4.5.2 User Engagement and Retention

- Challenge: Getting users to keep coming back and stay active on the platform.
- Solution: Added fun features like badges and leaderboards to make participation exciting. Used AI to suggest personalized content to keep users interested.

4.5.3 Cross-platform Compatibility

- Challenge: Making sure the platform works well on different devices and browsers.
- Solution: Focused on making a design that looks good on mobile first and used techniques to make sure it adjusts to any screen size or device.

4.5.4 Real-time Communication

- Challenge: Adding features for instant chatting and notifications.
- Solution: Used technology called WebSocket for live chatting and set up notifications to alert users about new messages or updates.

4.5.5 Content Moderation

- Challenge: Keeping the platform free from bad or harmful content posted by users.
- Solution: Made tools that automatically check content and used machine learning. Set rules for the community and let users report bad content.

4.5.6 Integration with University Systems

- Challenge: Connecting our platform with the university's existing systems and data securely.

- Solution: Worked with university IT teams to make sure the connection is secure. Used proper ways to handle login and data sharing that fit university privacy rules.

By tackling these challenges with practical answers, our team made sure the social network platform was ready to go and continued to be a great tool for university students.

4.6 Testing and Feedback

We made sure to thoroughly test our social network platform to make sure it works well, is reliable, and makes users happy.

4.6.1 Unit Testing

- We checked each part or piece of code to make sure it's doing its job right.
- We used tools like Jest for JavaScript to run these tests automatically and make sure everything is working as it should.

4.6.2 Integration Testing

- We tested to see if different parts of the app work well together.
- We pretended to be real users to see how the different parts interact and pass information.

4.6.3 System Testing

- We looked at the whole software, all put together, to make sure it does what it's supposed to.
- We checked everything from start to finish to make sure it meets all our requirements.

4.6.4 User Acceptance Testing (UAT)

- We let a group of actual users check if the system does what they need and want.

- This helped us see how real people would use the platform and if they like it.

4.6.5 Performance Testing

- We tested how fast and stable the app is under different situations.
- This included checking how the system handles normal and really busy times.

4.6.6 Security Testing

- We checked for any weak spots to make sure user data and the system are safe.
- We looked for common security problems using special tests and tools.

4.6.7 Usability Testing

- We had real users or experts test how easy the platform is to use.
- We focused on making sure the platform is easy to get around, understand, and use.

4.6.8 Feedback Collection

- We kept asking for feedback through surveys, talking to users, and watching how they use the platform.
- We made it easy for users to tell us what they think right on the platform, and we listened to what they had to say in group discussions.
- The feedback helped us see what's working and what's not. People liked the platform's simplicity, design, and how it fits with school life. We took any criticism or ideas for making things better seriously and made changes to better meet user needs.

This whole cycle of testing and getting feedback helped us make sure the platform is just right for what users need and expect from something made for the university community.

5 Results and Discussion

This part talks about how we can make our social network for the university better. We need the help of the university and need to use their tools and departments to do this.

5.1 Improvement of the Platform

5.1.1 Mobile App

We should make a mobile app to make it easy for everyone to use the platform on their phones. This app should send messages directly to users and work well on mobile devices. Also, it should work smoothly with the university's systems like online learning and library databases to make everything easier for users.

5.1.2 Platform Usability

We should change the platform to show users things they like or are interested in, based on their classes or what they do on the platform. We should also add different languages so more students can use it comfortably.

5.2 Helping Community and Sharing

5.2.1 Making It Easier to Share and Create Content

We should make it simpler for people to put up their own posts, stories, and educational materials. This makes the platform richer and helps everyone help each other out.

5.2.2 Getting Feedback and Helping Each Other

We need a better way for users to tell us what they think and what they need from the platform. We also should create places like forums and events where users can meet and help each other.

5.3 Keeping the Platform Good

5.3.1 Always Watching and Updating

We need to keep an eye on the platform and update it based on what users say and what we find from the data. This helps us fix problems quickly and add new things that users want.

5.3.2 Working with the University

We need ongoing help from the university, like resources and working together with different departments, to keep making the platform better.

In short, we are suggesting some changes to make our platform better for everyone at the university. We want to make the platform easy to use on phones, more personal for users, and better at bringing the community together. We also want to make sure we keep updating the platform and work closely with the university to do this.

6 Conclusion

In summary, the creation and introduction of the social network platform specifically designed for our university community have been overwhelmingly positive, achieving the main goals we set out at the beginning. This platform has been particularly crucial during times when remote learning and social distancing were necessary, providing a much-needed space for communication and collaboration within our academic setting.

Key accomplishments include:

- **Enhanced Communication:** There has been a significant improvement in how students, faculty, and staff communicate with each other. This has led to better collaboration and sharing of knowledge, which has made our academic community more connected.

- **Community Building:** The platform has helped build a stronger sense of unity and belonging among all members of the university, making our educational environment more inclusive and supportive.

- **Resource Sharing:** It has turned into an essential tool for distributing educational content, research results, and information about academic opportunities, which has enriched our learning experiences and contributed to academic achievement.

- **Increased Engagement:** We've seen a clear rise in how much and how often users participate in discussions, events, and other activities on the platform, showing that the community is more active and involved.

- **Feedback and Improvement:** Ongoing feedback from users has been invaluable, allowing us to continually refine and enhance the platform to better meet the needs of our community.

Overall, this social network platform has proven to be an effective tool in enhancing the educational journey, fostering community involvement, and supporting the broader goals of the university. The development of this platform has established a solid base for continuous innovation and adjustment to meet the changing demands of our academic community. The success of this project highlights the critical role that targeted digital solutions can play in today's educational landscapes and suggests that similar strategies could have a significant positive impact on other institutions and communities as well. This thesis has not only contributed to our understanding of digital solutions in academic settings but also set a precedent for future projects aimed at improving educational environments through technology.

7. References

1. Le Vie, Jr., Donald. “*An eCommerce Primer forTechnical Communicators,*” STC Proceedings of the 47th Annual Conference, 2000.
2. Perry, Greg Sams. “*Teach Yourself BeginningProgramming in 24 Hours*”, Sams Publishing, 1998 , p.92
3. Najam Nazar, Aldeida Aleti, Yaokun Zheng.“*Feature-based software design pattern detection”* [online] . December 2021. https://reader.elsevier.com/reader/sd/pii/S0164121221002624?token=B3F437B558906EFA2239C679C95F7C89F2A215DB5964DCD20E85A47824C23F27A899A1BFD87631B318C95FBAA8F91709&originRegion=eu-west-1&originCreation=20220221230228
4. Alexander Christopher. “*Software Architecture Design Pattern in Java. 2004*”, p.23
5. Glenn E. Krasner Stephen T. Pope. “*A Cookbook for Using View-Controller User the ModelInterface Paradigm in Smalltalk-80*”. 1989 p.26-30
6. K.E. Depew. Social media at Academia’s periphery: “*Studying multilingual developmental writers facebook composing strategies*”. Reading Matrix: An International Online Journal, 11 (2011)
7. Adam Boduch. **“***Flux Architecture*”. May 2016. p93-115
8. Todd Abel.**“***ReactJS: Become a professional in web app development 2016*”, p.80-96
9. Jess Chadwick , Todd Snyder , Hrusikesh Panda. **“***Programming ASP.NET MVC 4: Developing Real-World Web Applications with ASP.NET MVC* ”. 2012, p87-93
10. Azat Mardan. **“***React Quickly: Painless Web Apps with React, JSX, Redux and GraphQL*”. 2017, p.275-278
11. Sangeeth Arulraj. Flux vs Redux [online]. March 2020. https://medium.com/nerd-for-tech/flux-vs-redux-6cb572f8d7f8
12. Paul McFedries. “*Web Coding & Development All-in-One for Dummies*”. 2018 p.197-198
13. Alex Banks & Eve Porcello. **“***Learning React: Functional Web Development with React and Flux*”. 2017, 14-16

14. TutorialsPoint. "*Download HTML Tutorial*". 2016, p.16
15. Nicholas C.Zakas. "*JavaScript for Web Developers*". 2012, p. 43-46
16. Luis Atencio. "*Functional Programming in JavaScript*". 2016, p. 117-144. ISBN: 9781617292828
17. Kirupa Chinnathambi. "*LEARNING REACT*".November 2016, p.81-88. ISBN-13: 978-0-134-54631-5
18. Ilya Gelman Boris Dinkevich. "**The Complete Redux Book**"[online]. 2017, p 112-134

8 List of figures and abbreviations

8.1 Figures

8.2 List of abbreviations

API - Application Programming Interface

BLL - Business Logic Layer

DAL Data - Access Layer

HTML - Hypertext Markup Language

JS - JavaScript

JSX - JavaScript XML

MVC - Model-View-Controller

IT - Information Technology

HOC - High Order Component